Cardinality of Equivalence Relations

Lukas Bulwahn

April 19, 2020

Abstract

This entry provides formulae for counting the number of equivalence relations and partial equivalence relations over a finite carrier set with given cardinality.

To count the number of equivalence relations, we provide bijections between equivalence relations and set partitions [4], and then transfer the main results of the two AFP entries, Cardinality of Set Partitions [1] and Spivey’s Generalized Recurrence for Bell Numbers [2], to theorems on equivalence relations. To count the number of partial equivalence relations, we observe that counting partial equivalence relations over a set \( A \) is equivalent to counting all equivalence relations over all subsets of the set \( A \). From this observation and the results on equivalence relations, we show that the cardinality of partial equivalence relations over a finite set of cardinality \( n \) is equal to the \( n+1 \)-th Bell number [3].

Contents

1 Cardinality of Equivalence Relations 2
   1.1 Bijection between Equivalence Relations and Set Partitions 2
      1.1.1 Possibly Interesting Theorem for HOL.Equiv-Relations 2
      1.1.2 Dedicated Facts for Bijection Proof 2
      1.1.3 Bijection Proof 3
   1.2 Finiteness of Equivalence Relations 4
   1.3 Cardinality of Equivalence Relations 4

2 Cardinality of Partial Equivalence Relations 5
   2.1 Definition of Partial Equivalence Relation 5
   2.2 Construction of all Partial Equivalence Relations for a Given
      Set 6
   2.3 Injectivity of the Set Construction 7
   2.4 Cardinality Theorem of Partial Equivalence Relations 8
1 Cardinality of Equivalence Relations

theory Card-Equiv-Relations
imports Card-Partitions Card-Partitions
Bell-Numbers-Spivey Bell-Numbers
begin

1.1 Bijection between Equivalence Relations and Set Partitions

1.1.1 Possibly Interesting Theorem for HOL.Equiv-Relations

This theorem was historically useful in this theory, but is now after some proof refactoring not needed here anymore. Possibly it is an interesting fact about equivalence relations, though.

lemma equiv-quotient-eq-quotient-on-UNIV:
assumes equiv A R
shows \( A \, // \, R = (UNIV \, // \, R) - \{\}\) proof
show \( UNIV \, // \, R - \{\} \subseteq A \, // \, R \) proof
fix X
assume X :\( \in \) \( UNIV \, // \, R - \{\} \)
from this obtain x where \( X = R \, \{\} \) and \( X \neq \{\} \)
by (auto elim!: quotientE)
from this have x :\( \in \) \( A \)
using (equiv A R: equiv-class-eq-iff by fastforce)
from this (X :\( \in \) \( R \, \{\} \)) show X :\( \in \) \( A \, // \, R \)
by (auto intro!: quotientI)
qed
next
show A :\( \subseteq \) \( UNIV \, // \, R - \{\} \)
proof
fix X
assume X :\( \in \) \( A \, // \, R \)
from this have X :\( \notin \) \( \{\} \)
using (equiv A R: in-quotient-imp-non-empty by auto)
moreover from (X :\( \in \) \( A \, // \, R \)) have X :\( \in \) \( UNIV \, // \, R \)
by (metis UNIV-I assms proj-Eps proj-preserves)
ultimately show X :\( \in \) \( UNIV \, // \, R - \{\} \) by simp
qed

1.1.2 Dedicated Facts for Bijection Proof

lemma equiv-relation-of-partition-of:
assumes equiv A R
shows \( \{\,(x, y). \exists X \in A \, // \, R. \, x \in X \land y \in X\}\) = R
proof

show \{ (x, y). \exists X \in A \parallel R. x \in X \land y \in X \} \subseteq R

proof

fix xy

assume xy \in \{ (x, y). \exists X \in A \parallel R. x \in X \land y \in X \}

from this obtain x y and X where \( xy = (x, y) \)
and \( X \in A \parallel R \) and \( x \in X \land y \in X \)
by auto

from \( X \in A \parallel R \) obtain \( z \) where \( X = R \parallel \{ z \} \)
by (auto elim: quotientE)

show \( xy \in R \)
using \( \langle xy \rangle = (x, y) ; \langle X = R \parallel \{ z \} \rangle \langle x \in X \rangle \langle y \in X \rangle \langle \text{equiv} A \ R \rangle \)
by (simp add: equiv-class-eq-iff quotientI)

moreover have \( x \in R \parallel \{ x \} \)
using \( \langle (x, y) \rangle \in R \parallel \langle \text{equiv} A \ R \rangle \)
by (meson equiv-class-eq-iff equiv-class-self)

moreover have \( y \in R \parallel \{ x \} \)
using \( \langle (x, y) \rangle \in R \parallel \langle \text{equiv} A \ R \rangle \) by simp

ultimately have \( \langle (x, y) \rangle \in \{ (x, y), \exists X \in A \parallel R. x \in X \land y \in X \} \)
by auto

from this show \( xy \in \{ (x, y), \exists X \in A \parallel R. x \in X \land y \in X \} \)
using \( \langle xy \rangle = (x, y) \) by simp

qed

qed

1.1.3 Bijection Proof

lemma bij-betw-partition-of:

bij-betw \( \langle R. \ A \parallel R \rangle \{ R. \ \text{equiv} A \ R \} \{ P. \ \text{partition-on A} \ \ P \} \)

proof (rule bij-betw-byWitness[where \( f' = \lambda P. \ \{ (x, y), \exists X \in P. \ x \in X \land y \in X \} \]))

show \( \forall R \in \{ R. \ \text{equiv} A \ R \}, \{ (x, y), \exists X \in A \parallel R. \ x \in X \land y \in X \} = R \)
by (simp add: equiv-relation-of-partition-of)

show \( \forall P \in \{ P. \ \text{partition-on A} \ \ P \}, \ A \parallel \{ (x, y), \exists X \in P. \ x \in X \land y \in X \} = P \)
by (simp add: partition-on-eq-quotient)

show \( \langle R. \ A \parallel R \rangle \cdot \{ R. \ \text{equiv} A \ R \} \subseteq \{ P. \ \text{partition-on A} \ \ P \} \)
using partition-on-quotient by auto

show \( \langle P. \ \{ (x, y), \exists X \in P. \ x \in X \land y \in X \} \rangle \cdot \{ P. \ \text{partition-on A} \ \ P \} \subseteq \{ R. \ \text{equiv} A \ R \} \)
by (auto elim: quotientE)

show \( \langle P. \ \{ (x, y), \exists X \in P. \ x \in X \land y \in X \} \rangle \cdot \{ R. \ \text{equiv} A \ R \} \subseteq \{ P. \ \text{partition-on A} \ \ P \} \)
using partition-on-quotient by auto

show \( \langle P. \ \{ (x, y), \exists X \in P. \ x \in X \land y \in X \} \rangle \cdot \{ R. \ \text{equiv} A \ R \} \subseteq \{ P. \ \text{partition-on A} \ \ P \} \)
by (auto elim: quotientE)
equiv A R
  using equiv-partition-on by auto
qed

lemma bij-betw-partition-of-equiv-with-k-classes:
bij_betw (\lambda R. A // R) \{ R. equiv A R ∧ card (A // R) = k \} \{ P. partition-on A P ∧ card P = k \}
proof (rule bij_betw_byWitness [where f' = \lambda P. \{ (x, y). \exists X \in P. x \in X ∧ y \in X \}])
show ∀ R ∈ \{ R. equiv A R ∧ card (A // R) = k \}. \{ (x, y). \exists X \in A // R. x \in X ∧ y \in X \} = R
  by (auto simp add: equiv-relation-of-partition-of)
show ∀ P ∈ \{ P. partition-on A P ∧ card P = k \}. A // \{ (x, y). \exists X \in P. x \in X ∧ y \in X \} = P
  by (auto simp add: partition-on-eq-quotient)
using partition-on-quotient by auto
show (\lambda P. \{ (x, y). \exists X \in P. x \in X ∧ y \in X \})' \{ P. partition-on A P ∧ card P = k \} = k
  \subseteq \{ R. equiv A R ∧ card (A // R) = k \}
using equiv-partition-on by (auto simp add: partition-on-eq-quotient)
qed

1.2 Finiteness of Equivalence Relations

lemma finite-equiv:
assumes finite A
shows finite \{ R. equiv A R \}
proof –
have bij_betw (\lambda R. A // R) \{ R. equiv A R \} \{ P. partition-on A P \}
  by (rule bij_betw_partition_of)
from this show finite \{ R. equiv A R \}
  using finite A \ finite_many_partition_on by (simp add: bij_betw_finite)
qed

1.3 Cardinality of Equivalence Relations

theorem card-equiv-rel-eq-card-partitions:
card \{ R. equiv A R \} = card \{ P. partition-on A P \}
proof –
have bij_betw (\lambda R. A // R) \{ R. equiv A R \} \{ P. partition-on A P \}
  by (rule bij_betw_partition_of)
from this show card \{ R. equiv A R \} = card \{ P. partition-on A P \}
  by (rule bij_betw_same_card)
qed

corollary card-equiv-rel-eq-Bell:
assumes finite A
shows card \{ R. equiv A R \} = Bell (card A)
using assms Bell_all_def card-equiv-rel-eq-card-partitions by force
corollary card-equiv-rel-eq-sum-Stirling:
  assumes finite A
  shows card \{ R \text{. equiv } A R \} = \sum (\text{Stirling } (\text{card } A)) \{ \text{..card } A \}
using assms card-equiv-rel-eq-Bell Bell-Stirling-eq by simp

theorem card-equiv-k-classes-eq-card-partitions-k-parts:
  card \{ R \text{. equiv } A R \land \text{card } (A // R) = k \} = card \{ P \text{. partition-on } A P \land \text{card } P = k \}
proof
  have bij-betw (λR A // R) \{ R \text{. equiv } A R \land \text{card } (A // R) = k \} \{ P \text{. partition-on } A P \land \text{card } P = k \}
    by (rule bij-betw-partition-of-equiv-with-k-classes)
  from this show card \{ R \text{. equiv } A R \land \text{card } (A // R) = k \} = card \{ P \text{. partition-on } A P \land \text{card } P = k \}
    by (rule bij-betw-same-card)
qed

corollary
  assumes finite A
  shows card \{ R \text{. equiv } A R \land \text{card } (A // R) = k \} = \text{Stirling } (\text{card } A) k
using card-equiv-k-classes-eq-card-partitions-k-parts
  card-partition-on \{ OF \text{..finite } A \} by metis
end

2  Cardinality of Partial Equivalence Relations

theory Card-Partial-Equiv-Relations
imports
  Card-Equiv-Relations
begin

2.1  Definition of Partial Equivalence Relation

definition partial-equiv := 'a set ⇒ ('a × 'a) set ⇒ bool
where
  partial-equiv A R = (R ⊆ A × A ∧ sym R ∧ trans R)

lemma partial-equivI:
  assumes R ⊆ A × A sym R trans R
  shows partial-equiv A R
using assms unfolding partial-equiv-def by auto

lemma partial-equiv iff:
  shows partial-equiv A R ⇔ (∃ A' ⊆ A. equiv A' R)
proof
  assume partial-equiv A R
  from ⟨partial-equiv A R⟩ have R " A ⊆ A
    unfolding partial-equiv-def by blast
moreover have equiv \((R \leftrightarrow A)\) \(R\)
proof (rule equivI)
  from \(\text{partial-equiv } A \, R\); show \(\text{sym } R\)
    unfolding partial-equiv-def by blast
  from \(\text{partial-equiv } A \, R\); show \(\text{trans } R\)
    unfolding partial-equiv-def by blast
show refl-on \((R \leftrightarrow A)\) \(R\)
proof (rule refl-onI)
  show \(R \subseteq R \leftrightarrow A \times R \leftrightarrow A\)
    proof
      fix \(p\)
      assume \(p \in R\)
      obtain \(x \, y\) where \(p = (x, y)\) by fastforce
      moreover have \(x \in R \leftrightarrow A\)
        using \(\langle \langle p \in R\rangle, \langle p = (x, y)\rangle, \langle \text{partial-equiv } A \, R\rangle \langle \text{sym-def} \rangle \rangle\) by fastforce
      moreover have \(y \in R \leftrightarrow A\)
        using \(\langle \langle p \in R\rangle, \langle p = (x, y)\rangle, \langle R \leftrightarrow A \subseteq A\rangle, \langle x \in R \leftrightarrow A\rangle \rangle\) by blast
    ultimately show \(p \in R \leftrightarrow A \times R \leftrightarrow A\) by auto
qed
next
fix \(y\)
assume \(y \in R \leftrightarrow A\)
from this obtain \(x\) where \((x, y) \in R\) by auto
from \(\langle (x, y) \in R\rangle\); have \((y, x) \in R\)
  using \(\langle \text{sym } R\rangle\) by (meson symE)
from \(\langle (x, y) \in R\rangle, \langle (y, x) \in R\rangle\); show \((y, y) \in R\)
  using \(\langle \text{trans } R\rangle\) by (meson transE)
qed
qed
ultimately show \(\exists A' \subseteq A. \, \text{equiv } A' \, R\) by blast
next
assume \(\exists A' \subseteq A. \, \text{equiv } A' \, R\)
from this obtain \(A'\) where \(A' \subseteq A\) and \(\text{equiv } A' \, R\) by blast
show partial-equiv \(A \, R\)
proof (rule partial-equivI)
  from \(\langle \text{equiv } A' \, R\rangle, \langle A' \subseteq A\rangle\); show \(R \subseteq A \times A\)
    using equiv-class-eq-iff by fastforce
  from \(\langle \text{equiv } A' \, R\rangle\); show \(\text{sym } R\)
    using equivE by blast
  from \(\langle \text{equiv } A' \, R\rangle\); show \(\text{trans } R\)
    using equivE by blast
qed
qed

2.2 Construction of all Partial Equivalence Relations for a Given Set

definition all-partial-equivs-on :: \(\text{'}a\, \text{set}\Rightarrow (\text{'}a \times \text{'}a)\, \text{set}\) set
where
all-partial-equivs-on $A =$
\[
\begin{align*}
& \text{do } \\
& \quad k \leftarrow \{0..\text{card } A\}; \\
& \quad A' \leftarrow \{A'. A' \subseteq A \land \text{card } A' = k\}; \\
& \quad \{R. \text{equiv } A' \} \\
\end{align*}
\]

\textbf{lemma} all-partial-equivs-on:
\textbf{assumes} finite $A$
\textbf{shows} all-partial-equivs-on $A = \{R. \text{partial-equiv } A R\}$
\textbf{proof}
\textbf{show} all-partial-equivs-on $A \subseteq \{R. \text{partial-equiv } A R\}$
\textbf{proof}
\textbf{fix} $R$
\textbf{assume} $R \in \text{all-partial-equivs-on } A$
\textbf{from} this \textbf{obtain} $A'$ \textbf{where} $A' \subseteq A$ \textbf{and} equiv $A' R$
\textbf{unfolding} all-partial-equivs-on-def \textbf{by} auto
\textbf{from} this \textbf{have} partial-equiv $A R$
\textbf{using} partial-equiv-iff \textbf{by} blast
\textbf{from} this \textbf{show} $R \in \{R. \text{partial-equiv } A R\}$ ..
\textbf{qed}
\textbf{next}
\textbf{show} $\{R. \text{partial-equiv } A R\} \subseteq \text{all-partial-equivs-on } A$
\textbf{proof}
\textbf{fix} $R$
\textbf{assume} $R \in \{R. \text{partial-equiv } A R\}$
\textbf{from} this \textbf{obtain} $A'$ \textbf{where} $A' \subseteq A$ \textbf{and} equiv $A' R$
\textbf{using} partial-equiv-iff \textbf{by} (metis mem-Collect-eq)
\textbf{moreover} \textbf{have} $\text{card } A' \in \{0..\text{card } A\}$
\textbf{using} (A' $\subseteq$ A. (finite A)) \textbf{by} (simp add: card-mono)
\textbf{ultimately} \textbf{show} $R \in \text{all-partial-equivs-on } A$
\textbf{unfolding} all-partial-equivs-on-def
\textbf{by} (auto simp del: atLeastAtMost-iff)
\textbf{qed}
\textbf{qed}

\textbf{2.3 Injectivity of the Set Construction}

\textbf{lemma} equiv-inject:
\textbf{assumes} equiv $A R$ equiv $B R$
\textbf{shows} $A = B$
\textbf{proof}
\textbf{from} \textbf{assms} \textbf{have} $R \subseteq A \times A R \subseteq B \times B$ \textbf{by} (simp add: equiv-type)+
\textbf{moreover from} \textbf{assms} \textbf{have} $\forall x \in A. (x, x) \in R \forall x \in B. (x, x) \in R$
\textbf{by} (simp add: eq-equiv-class)+
\textbf{ultimately} \textbf{show} ?thesis
\textbf{using} mem-Sigma-iff subset-antisym subset-eq \textbf{by} blast
\textbf{qed}
lemma injectivity:
assumes \((A' \subseteq A \land \text{card } A' = k) \land (A'' \subseteq A \land \text{card } A'' = k')\)
assumes equiv \(A' R \land equiv A'' R'\)
assumes \(R = R'\)
shows \(A' = A'' \land k = k'\)
proof
  from \(<R = R'>\) assms(2) show \(A' = A''\)
  using equiv-inject by fast
  from this assms(1) show \(k = k'\) by simp
qed

2.4 Cardinality Theorem of Partial Equivalence Relations

theorem card-partial-equiv:
assumes finite \(A\)
shows \(\text{card} \ \{R. \text{partial-equiv } A \ R\} = \text{Bell} (\text{card } A + 1)\)
proof
  let \(?expr = do\{
    k \leftarrow \{0..\text{card } A\};
    A' \leftarrow \{A'. \ A' \subseteq A \land \text{card } A' = k\};
    \{R. \text{equiv } A' \ R\\}
  \}\)
  have \(\text{card} \ \{R. \text{partial-equiv } A \ R\} = \text{card} \ (\text{all-partial-equivs-on } A)\)
    using \(<\text{finite } A>\) by \((\text{simp add: all-partial-equivs-on})\)
  also have \(\text{card} \ (\text{all-partial-equivs-on } A) = \text{card} \ ?expr\)
    unfolding all-partial-equivs-on_def ..
  also have \(\text{card} \ ?expr = \sum k = 0..\text{card } A. \ (\text{card } A \choose k) \ast \text{Bell } k\)
  proof
    let \(?S \gg = \text{?comp = ?expr}\)
    \{\fix k
      assume \(k \in \{0..\text{card } A\}\)
      let \(?expr = \text{?comp } k\)
      let \(?S \gg = \text{?comp = ?expr}\)
      have finite \(?S\) using \(<\text{finite } A>\) by simp
    moreover \{\fix A'
      assume \(A', A' \in \{A'. \ A' \subseteq A \land \text{card } A' = k\}\)
      from this have \(A' \subseteq A\) and \(\text{card } A' = k\) by auto
      let \(?expr = \text{?comp } A'\)
      have finite \(A'\)
        using \(<\text{finite } A>\) \(?expr \subseteq A\) by \((\text{simp add: finite-subset})\)
      have \(\text{card} \ ?expr = \text{Bell } k\)
        using \(<\text{finite } A>\) \(\text{finite } A' \land \text{card } A' = k\) by \((\text{simp add: card-equiv-rel-eq-Bell})\)
    moreover have finite \(?expr\)
      using \(<\text{finite } A'>\) by \((\text{simp add: finite-equiv})\)
    ultimately have finite \(\text{?expr} \land \text{card} \ ?expr = \text{Bell } k\) by blast
  qed
moreover have disjoint-family-on \( \comp \ ?S \)
   by (injectivity-solver rule: injectivity(1))
moreover have \( \text{card } \ ?S = \text{card } A \text{ choose } k \)
   using \( \text{finite } A \) by (simp add: \( n \)-subsets)
ultimately have \( \text{card } \ ?expr = (\text{card } A \text{ choose } k) \ast \text{Bell } k \) (is - = \( ?\text{formula} \))
   by (subst \( \text{card-bind-constant} \)) auto
moreover have \( \text{finite } \ ?expr \)
   using \( \text{finite } \ ?S \) by (intro: finite-bind)
ultimately have \( \text{finite } \ ?expr \land \text{card } \ ?expr = ?\text{formula} \) by blast
\}
moreover have \( \text{finite } \ ?S \) by simp
moreover have disjoint-family-on \( \comp \ ?S \)
   by (injectivity-solver rule: injectivity(2))
ultimately show \( \text{card } \ ?expr = (\sum_{k=0}^{\text{card } A} (\text{card } A \text{ choose } k) \ast \text{Bell } k) \)
   by (subst \( \text{card-bind} \)) auto
qed
also have \( \ldots = (\sum_{k \leq \text{card } A} (\text{card } A \text{ choose } k) \ast \text{Bell } k) \)
   by (auto intro: sum.cong)
also have \( \ldots = \text{Bell } (\text{card } A + 1) \)
   using \( \text{Bell-recursive-eq} \) by simp
finally show \( ?\text{thesis} \).
qed

References


